

# HIGH-SPEED LOOKUP TABLE CIRCUITS AND METHODS FOR PROGRAMMABLE LOGIC DEVICES

## FIELD OF THE INVENTION

**[0001]** The invention relates to Programmable Logic Devices (PLDs). More particularly, the invention relates to high-speed lookup table circuits and methods for PLDs.

## BACKGROUND OF THE INVENTION

**[0002]** Programmable logic devices (PLDs) are a well-known type of integrated circuit that can be programmed to perform specified logic functions. One type of PLD, the field programmable gate array (FPGA), typically includes an array of configurable logic blocks (CLBs) and programmable input/output blocks (IOBs). The CLBs and IOBs are interconnected by a programmable interconnect structure. Some FPGAs also include additional logic blocks with special purposes (e.g., DLLs, RAM, processors, and so forth).

**[0003]** The interconnect structure, CLBs, IOBs, and other logic blocks are typically programmed by loading a stream of configuration data (bitstream) into internal configuration memory cells that define how the logic blocks and interconnect are configured. The configuration data can be read from memory (e.g., an external PROM) or written into the FPGA by an external device. The collective states of the individual memory cells then determine the function of the FPGA.

**[0004]** Programmable lookup tables are often included in PLDs. A lookup table is a circuit that provides an output value selected from among two or more stored values in response to a combination of input values. One method of implementing a lookup table (LUT) is to store  $2^n$  values in memory cells (e.g., configuration memory cells in an FPGA), where  $n$  is the number of input signals to the LUT. Each of the  $n$  input signals is then utilized to select only half of the remaining stored values. For example, a 4-input LUT can

utilize the four input signals to select first eight, then four, then two, and finally one of 16 stored values.

**[0005]** Fig. 1 illustrates a well-known LUT circuit that functions according to this well-known method. A 2-input LUT is shown, as in many of the drawings herein, to clarify the drawings. Many LUTs now included in FPGAs, for example, utilize four input signals. The illustrated examples are easily modified to accommodate four input signals (or other numbers of input signals) by one of skill in the relevant arts.

**[0006]** The LUT circuit of Fig. 1 has two input signals IN1 and IN2, and can provide at the output terminal OUT any of four values stored in memory cells MC-0 through MC-3. (In the present specification, the same reference characters are used to refer to terminals, signal lines, and their corresponding signals.) The LUT circuit includes six inverters 101-106, six N-channel transistors 111-116, and a pullup 121 implemented using a P-channel transistor. N-channel transistors 111-116 implement a well-known 4-to-1 multiplexer circuit.

**[0007]** Inverters 105-106 and pullup 121 implement a well-known output buffer. The output (node M) of the multiplexer circuit drives inverter 105 and is pulled to power high (VDD) by P-channel transistor 121. The output terminal of inverter 105 is coupled to the gate terminal of pullup 121 and also drives inverter 106, which provides the output signal OUT to the output terminal. Pullup 121 serves to pull node M fully to power high VDD when a memory cell storing a high value is selected by transistors 111-116.

**[0008]** The LUT circuit of Fig. 1 functions as shown in the example illustrated in Fig. 2, wherein both input signals IN1 and IN2 are high, memory cells MC-0 through MC-2 store low values, and memory cell MC-3 stores a high value. When input signal IN2 is high, the output of inverter 101 goes low, turning off transistors 112 and 114. The output of inverter 101 also drives inverter 103, so the output of inverter 103

goes high, turning on transistors 111 and 113. Similarly, when input signal IN1 is high, the output of inverter 102 goes low, turning off transistor 116. The output of inverter 102 also drives inverter 104, so the output of inverter 104 goes high, turning on transistor 115. Thus, a path is enabled between memory cell MC-3 and node M, and the high value stored in memory cell MC-3 is provided to node M and hence to output terminal OUT. A heavy line in Fig. 2 shows the signal path now enabled between memory cell MC-3 and output terminal OUT.

**[0009]** Another well-known LUT circuit (not shown) is similar to that of Fig. 1. N-channel transistors 111-116 are replaced by CMOS passgates, which include N-channel and P-channel transistors coupled in parallel. The N-channel transistors are coupled as shown in Fig. 1. Each P-channel transistor is gated by the complementary signal to the signal driving the associated N-channel transistor. In this version of the circuit pullup 121 can be omitted, because the P-channel transistors in the CMOS passgates ensure a fully high value on node M.

**[0010]** A PLD can include hundreds of thousands of LUTs implementing large amounts of user logic. In some designs, several LUTs are preferably included on each path between two clocked elements. Hence, the delay through each LUT can be quite significant in determining the speed at which a user design will operate. Therefore, it is desirable to provide a LUT circuit having a reduced delay when compared to known LUT circuits.

#### SUMMARY OF THE INVENTION

**[0011]** The invention provides lookup table (LUT) circuits and methods that can be used, for example, in programmable logic devices (PLDs). A LUT according to one embodiment comprises a multiplexer circuit having two modes. In a first mode, the multiplexer circuit functions as a standard multiplexer, e.g., each select input signal is used to select

only half of the remaining stored values, until a single stored value is selected as the multiplexer output. In a second mode, the multiplexer circuit selects two or more stored values, where the selected values all have the same logical value. All of the selected values are provided to the multiplexer output terminal. The effective resistance through the multiplexer is reduced. Thus, in the second mode the multiplexer circuit provides a more rapid propagation of the stored values to the output terminal.

**[0012]** In some embodiments, the mode of the multiplexer circuit is controlled by a programmed value stored in a memory cell, e.g., a configuration memory cell in a PLD. In one embodiment, two select terminals of a multiplexer circuit are coupled to two different signal lines. When both signal lines are used in the PLD design, the multiplexer circuit is placed into the first mode, i.e., the multiplexer circuit functions as a standard multiplexer, selecting one output value based on signals provided by the first and second signal lines. When only one of the signal lines is used, the multiplexer circuit is placed into the second mode, in which a value on the unused signal line is ignored, and two identical stored values are provided to the output terminal. Thus, the multiplexer circuit has a reduced path delay when one of the two signal lines is unused.

#### BRIEF DESCRIPTION OF THE DRAWINGS

**[0013]** The present invention is illustrated by way of example, and not by way of limitation, in the following figures.

**[0014]** Fig. 1 shows a well-known lookup table (LUT) circuit.

**[0015]** Fig. 2 provides an example of the functionality of the LUT circuit of Fig. 1.

**[0016]** Fig. 3 shows a first 2-input LUT circuit according to an embodiment of the present invention.

**[0017]** Figs. 4A-4C provide examples of the functionality of the LUT circuit of Fig. 3.

**[0018]** Fig. 5 shows a second 2-input LUT circuit according to an embodiment of the present invention.

**[0019]** Fig. 6 provides an example of the functionality of the LUT circuit of Fig. 5.

**[0020]** Fig. 7 shows a third 2-input LUT circuit according to an embodiment of the present invention.

**[0021]** Fig. 8 provides an example of the functionality of the LUT circuit of Fig. 7.

**[0022]** Fig. 9 shows a fourth 2-input LUT circuit according to an embodiment of the present invention.

**[0023]** Fig. 10 provides an example of the functionality of the LUT circuit of Fig. 9.

**[0024]** Fig. 11 shows a first 4-input LUT circuit according to an embodiment of the present invention.

**[0025]** Fig. 12 shows a second 4-input LUT circuit according to an embodiment of the present invention.

**[0026]** Fig. 13 shows a third 4-input LUT circuit according to an embodiment of the present invention.

**[0027]** Fig. 14 shows a fourth 4-input LUT circuit according to an embodiment of the present invention.

**[0028]** Fig. 15 illustrates the steps of a method of providing a logical value, according to an embodiment of the present invention.

**[0029]** Fig. 16 illustrates the steps of a method of providing a logical value in a PLD, according to an embodiment of the present invention.

**[0030]** Fig. 17 illustrates the steps of a method of providing a logical value from a LUT, according to an embodiment of the present invention.

#### DETAILED DESCRIPTION OF THE DRAWINGS

**[0031]** The present invention is believed to be applicable to a variety of programmable logic devices (PLDs). The

present invention has been found to be particularly applicable and beneficial for field programmable gate arrays (FPGAs). While the present invention is not so limited, an appreciation of the present invention is presented by way of specific examples, in this instance with a variety of lookup table (LUT) circuits utilizing the configuration memory cells of an FPGA.

**[0032]** Fig. 3 illustrates a LUT circuit for an FPGA. To clarify the drawing, a 2-input LUT is shown, as in many of the drawings herein. However, the principles of the invention can be applied to LUTs having any number of input signals, e.g., four, six, or any other number greater than one. The illustrated examples are easily modified to accommodate other numbers of input signals by one of skill in the relevant arts.

**[0033]** The LUT circuit of Fig. 3 has two input signals IN1 and IN2, and can provide at the output terminal OUT any of four values stored in memory cells MC-0 through MC-3. The illustrated circuit includes four logical NAND gates 301-304, two inverters 105-106, six N-channel transistors 111-116, and a pullup 121 implemented (in the pictured embodiment) using a P-channel transistor. Elements 105-106, 111-116, 121, and MC-0 through MC-3 can be implemented and interconnected, for example, as in the known LUT circuit of Fig. 1. Therefore, these elements are not again described herein. Memory cells MC-A and MC-B can be, for example, configuration memory cells in an FPGA. In other embodiments, memory cells MC-A and MC-B are other storage elements such as flip-flops. In yet other embodiments, memory cells MC-A and MC-B are not included in the circuit, and the signals provided by these memory cells in the illustrated embodiments are instead provided by source(s) external to the LUT circuit.

**[0034]** The multiplexer circuit comprising N-channel transistors 111-116 is controlled in Fig. 3 by four select signals, each select signal being provided by a different one of logical NAND gates 301-304. Each logical NAND gate has

one input terminal coupled to one of memory cells MC-A and MC-B. Logical NAND gate 301 is driven by input signal IN2 and memory cell MC-A. Logical NAND gate 302 is driven by input signal IN1 and memory cell MC-B. Logical NAND gate 303 is driven by logical NAND gate 301 and memory cell MC-A. Finally, logical NAND gate 304 is driven by logical NAND gate 302 and memory cell MC-B.

**[0035]** (The term "logical NAND gate" is used herein to designate a logical circuit performing a NAND function. The most common form of a logical NAND circuit is, of course, a NAND gate. However, as is known to those of skill in the art, a logical NAND circuit can also be implemented in other ways, e.g., as an OR gate with inverted inputs. Similarly, the term "inverting logic gate" can refer to an inverter, but can also refer to another circuit that provides an inverting function, e.g., a two-input logic gate having an enable input as a second input.)

**[0036]** The LUT circuit of Fig. 3 functions, for example, as shown in Figs. 4A-4C.

**[0037]** In the example illustrated in Fig. 4A, a high value is stored in both of memory cells MC-A and MC-B. In this example, clearly the circuit becomes logically equivalent to the circuit shown in Fig. 1. Therefore, signals IN1 and IN2 are both used to select one of the four values stored in memory cells MC-0 through MC-3. A heavy line in Fig. 4A shows the signal path enabled between memory cell MC-3 and output terminal OUT. In this example, the LUT circuit functions in the same manner as that shown in Fig. 2.

**[0038]** In the example illustrated in Fig. 4B, input signal IN2 is high, input signal IN1 has an unknown value (X, or a "don't care" value), memory cell MC-A stores a high value, and memory cell MC-B stores a low value. In the pictured example, memory cells MC-0 and MC-2 store low values, and memory cells MC-1 and MC-3 store high values. However, in another embodiment (not shown) memory cells MC-0 and MC-2 store high values, and memory cells MC-1 and MC-3 store low

values. In the exemplary scenario of Fig. 4B, logical NAND gate 301 provides a low value, because signal IN2 is high and memory cell MC-A stores a high value. Logical NAND gate 303 provides a high value, because of the low value provided by logical NAND gate 301. Thus, the first stage of the multiplexer circuit (N-channel transistors 111-114) serves to select two of the four stored values, based on the value of input signal IN2.

**[0039]** The second stage of the multiplexer circuit (N-channel transistors 115-116) is controlled by logical NAND gates 302 and 304. Each of gates 302 and 304 provides a high value, irregardless of the value of input signal IN1, because of the low value stored in memory cell MC-B. Thus, both paths through the second stage are enabled, and the stored values from both of memory cells MC-3 and MC-1 are passed to node M, and hence to output terminal OUT of the LUT circuit. Heavy lines in Fig. 4B show the two signal paths now enabled between memory cell MC-3 and output terminal OUT, and between memory cell MC-1 and output terminal OUT. Because two "copies" of the stored value are provided to node M, node M assumes the stored value more rapidly than in the example of Fig. 4A. The effective resistance through the multiplexer is reduced. Thus, the delay through the LUT circuit is reduced.

**[0040]** Clearly, when two stored values are provided to the same node (node M), the two stored values must have the same logical value. In other words, both values must be high, or both values must be low. In the embodiment of Figs. 3 and 4A-4C, memory cells MC-0 through MC-3 are configuration memory cells of an FPGA. Thus, in the embodiment of Fig. 4B, memory cells MC-3 and MC-1 are configured with the same logical value, and memory cells MC-2 and MC-0 are configured with the same logical value. Either of the two logical values can be selected, depending on the value of input signal IN2.

**[0041]** In the example illustrated in Fig. 4C, input signal IN1 is high, input signal IN2 has an unknown value, memory



cell MC-A stores a low value, and memory cell MC-B stores a high value. In the pictured example, memory cells MC-0 and MC-1 store low values, and memory cells MC-2 and MC-3 store high values. However, in another embodiment (not shown) memory cells MC-0 and MC-1 store high values, and memory cells MC-2 and MC-3 store low values.

**[0042]** In the exemplary scenario of Fig. 4C, each of NAND gates 301 and 303 provides a high value, irregardless of the value of input signal IN2, because of the low value stored in memory cell MC-A. Thus, all four of the stored values are passed through to the second stage of the multiplexer circuit, with two stored values (from memory cells MC-2 and MC-3) being provided to node P and two stored values (from memory cells MC-0 and MC-1) being provided to node Q. Because two "copies" of the stored values are provided to each of nodes P and Q, the effective resistance between the memory cells and node M is lower than in the example of Fig. 4A. Thus, the delay through the LUT circuit is reduced.

**[0043]** In the second stage of the multiplexer circuit (N-channel transistors 115-116), logical NAND gate 302 provides a low value, because signal IN1 is high and memory cell MC-B stores a high value. Logical NAND gate 304 provides a high value, because of the low value provided by logical NAND gate 302. Thus, the second stage of the multiplexer circuit selects one signal that represents two of the four stored values, based on the value of input signal IN1. Heavy lines in Fig. 4C show the two signal paths now enabled between memory cell MC-3 and output terminal OUT, and between memory cell MC-2 and output terminal OUT.

**[0044]** As previously noted in connection with Fig. 4B, when two stored values are provided to the same node (node M), the two stored values must have the same logical value. In other words, both values must be high, or both values must be low. In the embodiment of Figs. 3 and 4A-4C, memory cells MC-0 through MC-3 are configuration memory cells of an FPGA. Thus, in the embodiment of Fig. 4C, memory cells MC-3 and MC-

2 are configured with the same logical value, and memory cells MC-1 and MC-0 are configured with the same logical value. Either of the two logical values can be selected, depending on the value of input signal IN1.

**[0045]** Clearly, the assignment of logical values to memory cells MC-0 through MC-3 is dependent on which signal paths through the multiplexer circuit can be enabled at the same time, which in turn depends on the values stored in memory cells MC-A and MC-B. This principle also holds true when the examples of Figs. 3 and 4A-4C are extended to larger multiplexer circuits having larger numbers of memory cells and larger numbers of signal paths.

**[0046]** In some embodiments, in which the LUT circuit forms a portion of a PLD, input terminal IN1 is coupled to a first signal line, and input terminal IN2 is coupled to a second signal line. When both signal lines are in use (i.e., included in a user design programmed into the PLD), as shown in Fig. 4A, the PLD is configured to store a high value in each of memory cells MC-A and MC-B. Because two signal lines are in use (i.e., two select signals are provided), four stored values are provided and each stored value can have any logical value. When only the second signal line is in use (i.e., the signal line coupled to terminal IN2), as shown in Fig. 4B, the PLD is configured to store a high value in memory cell MC-A and a low value in memory cell MC-B, and two stored values are provided to node M. When only the first signal line is in use (i.e., the signal line coupled to terminal IN1), as shown in Fig. 4C, the PLD is configured to store a low value in memory cell MC-A and a high value in memory cell MC-B, and two stored values are provided to node M.

**[0047]** Clearly, including the LUT circuit of Fig. 3 in a PLD can provide shorter LUT delays when some LUT input terminals remain unused (i.e., not utilized in a user design implemented in the PLD). It is common for many LUT input terminals to remain unused, particularly when the LUTs have

four input terminals or even larger numbers of input terminals (e.g., six or eight). Therefore, the LUT circuit of Fig. 3 (and the other exemplary LUT circuits shown and/or described herein) could significantly reduce circuit delays in PLDs.

**[0048]** As previously noted with regard to Figs. 4B and 4C, an unused LUT input terminal can have a "don't-care" value. In other words, when an appropriate value is stored in the memory cell associated with the input terminal, any signal value on the input terminal is ignored. (Note that while a low value in the memory cell causes the input signal value to be ignored in the pictured embodiments, a high value can have the same function in other embodiments.) However, in some PLDs unused signal lines have a known value, and this characteristic can be used to simplify the LUT circuit.

**[0049]** Fig. 5 illustrates a LUT circuit according to an embodiment of the invention in which unused signal lines have a low value. The circuit of Fig. 5 is similar to that of Fig. 3, except that logical NAND gates 301 and 302 are replaced by inverting logic gates (e.g., inverters) 501 and 502, which are driven by input signals IN2 and IN1, respectively. Because unused signal lines always have a low value in this embodiment, the LUT circuit of Fig. 5 provides the same logical functionality as the circuit of Fig. 3.

**[0050]** When both input terminals are coupled to used interconnect lines, high values are programmed into both of memory cells MC-A and MC-B. Thus, logical NAND gates 303 and 304 are equivalent to inverters 103 and 104 of Fig. 1, and the LUT circuit functions in the same fashion as the circuit of Fig. 1.

**[0051]** When only one of the input terminals is coupled to a used signal line, a low value is programmed into the memory cell associated with the unused input terminal. For example, Fig. 6 illustrates the scenario when a first signal line coupled to input terminal IN1 is unused, and a second signal line coupled to input terminal IN2 is used. In the pictured

embodiment, a high value is provided on signal line IN2. Thus, inverting logic gate 501 provides a low signal and logical NAND gate 303 provides a high signal, as in Fig. 4. Signal line IN1 provides a low signal, because the signal line is unused (U). Thus, inverting logic gate 502 and logical NAND gate 304 each provide a high signal, selecting both paths through the second stage of the multiplexer circuit, as in the embodiment of Fig. 4. Thus, the circuit of Fig. 5 behaves in the same fashion as described in connection with Fig. 3, as long as an unused signal line always has a low value.

**[0052]** Fig. 7 shows another modification of the circuit of Fig. 3, in which N-channel transistors 111-116 are replaced by CMOS passgates 711-716. The N-channel gate terminals of the CMOS passgates are coupled in the same fashion as the gate terminals of the corresponding N-channel transistors in Fig. 3. The output signal from logical NAND gate 301 is inverted by inverting logic gate (e.g., inverter) 707 and provided to the P-channel gate terminals of CMOS passgates 712 and 714. Similarly, the output signal from logical NAND gate 302 is inverted by inverting logic gate (e.g., inverter) 708 and provided to the P-channel gate terminal of CMOS passgate 716.

**[0053]** In one embodiment (not shown), the output signals from logical NAND gates 303 and 304 are similarly inverted and provided to the P-channel gate terminals of CMOS passgates 711 and 713, and 715, respectively. However, in the illustrated embodiment the LUT circuit is simplified by providing the output signals from logical NAND gates 301 and 302 to the P-channel gate terminals of CMOS passgates 711 and 713, and 715, respectively. The two embodiments are not completely equivalent, because at times one of the P-channel gates can be turned off while the corresponding N-channel gate is turned on (e.g., see passgate 715 in Fig. 8). However, this situation only occurs when two signal paths are enabled through the multiplexer circuit, so the P-channel

transistor in the other of the two enabled paths performs the function of pulling node M fully to power high VDD. Thus, pullup 121 is not needed in this circuit. In other respects, the circuit of Fig. 7 has the same functionality as the circuit of Fig. 3, e.g., as can be seen by comparing the examples shown in Figs. 8 and 4.

**[0054]** Fig. 9 shows another embodiment of the invention, in which the LUT circuit of Fig. 5 is modified in a fashion similar to the way in which Fig. 7 was modified from Fig. 3. N-channel transistors 111-116 are replaced by CMOS passgates 711-716, as in Fig. 7. The N-channel gate terminals of the CMOS passgates are coupled in the same fashion as the gate terminals of the corresponding N-channel transistors in Fig. 5. Input signals IN2 and IN1 are provided to the P-channel gate terminals of CMOS passgates 712 and 714, and 716, respectively.

**[0055]** In one embodiment (not shown), the output signals from logical NAND gates 303 and 304 are inverted and provided to the P-channel gate terminals of CMOS passgates 711 and 713, and 715, respectively. However, in the illustrated embodiment the LUT circuit is simplified by providing the output signals from inverting logic gates 501 and 502 to the P-channel gate terminals of CMOS passgates 711 and 713, and 715, respectively. Again, the two embodiments are not completely equivalent, because at times one of the P-channel gates can be turned off while the corresponding N-channel gate is turned on (e.g., see passgate 715 in Fig. 10). However, as in the embodiment of Fig. 7, in this situation two signal paths are enabled through the multiplexer circuit, so the P-channel transistor in the other of the two enabled paths performs the function of pulling node M fully to power high VDD. Thus, pullup 121 is not needed in this circuit. In other respects, the circuit of Fig. 9 has the same functionality as the circuit of Fig. 5, e.g., as can be seen by comparing the examples shown in Figs. 10 and 7.

**[0056]** Several embodiments have been shown that constitute variations on the basic LUT circuit shown in Fig. 3. Clearly, many more variations are possible. For example, active-high signals can be replaced with active-low signals by making straightforward alterations to the circuitry, such as are well known in the art of circuit design. Logical circuits can be replaced by their logical equivalents by appropriately inverting input and output signals, as is also well known. The number of stored values and input (select) signals can be increased or decreased, along with the number of multiplexer stages. Multiplexer implementations other than the simple illustrated examples can be used. For example, the N-channel transistors and CMOS passgates illustrated in the examples herein can be replaced by other logic elements coupled to the select terminals of the multiplexer. It will be apparent to one skilled in the art after reading this specification that the present invention can be practiced within these and other architectural variations.

**[0057]** Fig. 11 illustrates an exemplary LUT circuit having four input signals IN1-IN4, and providing at the output terminal OUT any of sixteen values stored in memory cells MC-0 through MC-15. The LUT circuit includes nine inverting logic gates (e.g., inverters) 1101-1109, four logical NAND gates 1111-1114, twenty memory cells MC-0 through MC-15 and MC-A through MC-D, thirty N-channel transistors 1131-1160, and pullups 1121-1125 implemented (in the pictured embodiment) using P-channel transistors. N-channel transistors 1131-1160 implement a well-known 16-to-1 multiplexer circuit, the output of which (node M) drives inverting logic gate 1109 and is pulled to power high (VDD) by P-channel transistor 1121. The output terminal of inverting logic gate 1109 is coupled to the gate terminal of pullup 1121 and also provides the output signal OUT to the output terminal. Pullup 1121 serves to pull node M fully to power high VDD when a memory cell storing a high value is

selected by transistors 1131-1160. Inverting logic gates 1105-1108 and pullups 1122-1125 perform a similar function halfway through each signal path traversing the multiplexer circuit.

**[0058]** The LUT circuit of Fig. 11 functions in a fashion similar to that described in conjunction with Fig. 5, except for an increased number of stages with the multiplexer. Note also that the number of enabled signal paths through the multiplexer circuit in the pictured embodiment can be as high as eight, depending on the number of unused input terminals. For example, suppose that input terminals IN1-IN3 are all unused. Input signal IN1 enables two paths, input signal IN2 increases the number of enabled paths to four, and input signal IN3 increases the number of enabled paths to eight. Such a reduction could significantly reduce the delay through the multiplexer circuit. As noted above, the same logical value must be stored at all locations from which signal paths are simultaneously enabled to the output terminal of the multiplexer circuit.

**[0059]** Clearly, the circuit of Fig. 11 can be easily extended yet further to accommodate even larger numbers of input signals and stored values, by adding additional stages to the multiplexer circuit. For example, one embodiment of the invention has six input signals and enables one, two, four, eight, sixteen, or thirty-two of sixty-four different signal paths through the multiplexer circuit.

**[0060]** Fig. 12 illustrates an embodiment in which the LUT circuit of Fig. 11 is modified in a fashion similar to the way in which Fig. 9 was modified from Fig. 5. N-channel transistors 1131-1160 are replaced by CMOS passgates 1231-1260. The N-channel gate terminals of the CMOS passgates are coupled in the same fashion as the gate terminals of the corresponding N-channel transistors in Fig. 11. Input signal IN4 is provided to the P-channel gate terminals of CMOS passgates 1232, 1234, 1236, 1238, 1240, 1242, 1244, and 1246. Input signal IN3 is provided to the P-channel gate terminals

of CMOS passgates 1248, 1250, 1252, and 1254. Input signal IN2 is provided to the P-channel gate terminals of CMOS passgates 1256 and 1258. Input signal IN1 is provided to the P-channel gate terminal of CMOS passgate 1260.

**[0061]** In one embodiment (not shown), the output signal from logical NAND gate 1111 is inverted and provided to the P-channel gate terminals of CMOS passgates 1231, 1233, 1235, 1237, 1239, 1241, 1243, and 1245. Similarly, the output signal from logical NAND gate 1112 is inverted and provided to the P-channel gate terminals of CMOS passgates 1247, 1249, 1251, and 1253; the output signal from logical NAND gate 1113 is inverted and provided to the P-channel gate terminals of CMOS passgates 1255 and 1257; and the output signal from logical NAND gate 1114 is inverted and provided to the P-channel gate terminal of CMOS passgate 1259.

**[0062]** In the pictured embodiment the LUT circuit is simplified by providing the output signals from the existing inverting logic gates to the P-channel gate terminals of the CMOS passgates, in a fashion similar to that employed in the embodiment of Fig. 9. The two embodiments are not completely equivalent, because at times one of the P-channel gates can be turned off while the corresponding N-channel gate is turned on. However, as previously noted, as long as at least one path through the multiplexer circuit provides enabled P-channel gates along its full length, pullups 1121-1125 (see Fig. 11) are not needed in this circuit.

**[0063]** Fig. 13 illustrates another variation on the circuit of Fig. 11. In this embodiment, the capabilities provided by the present invention are provided to some of the input terminals of the circuit, and not to other input terminals. Note that, comparing Fig. 13 to Fig. 11, memory cells MC-C and MC-D are included, while memory cells MC-A and MC-B are omitted. Logical NAND gates 1111 and 1112 are replaced by inverting logic gates (e.g., inverters) 1311 and 1312. This embodiment might be used, for example, if the heavy loading on logical NAND gates 1111 and 1112 (see Fig.



11) proves to slow down the circuit enough to counteract the speed benefits provided by the invention.

**[0064]** Fig. 14 illustrates another variation on the circuit of Fig. 11, in which the capabilities of the invention are added to alternating input terminals of the circuit. Note that, comparing Fig. 14 to Fig. 11, memory cells MC-B and MC-D are included, while memory cells MC-A and MC-C are omitted. Logical NAND gates 1111 and 1113 are replaced by inverting logic gates (e.g., inverters) 1411 and 1413.

**[0065]** Fig. 15 illustrates an exemplary method of providing a logical value, according to an embodiment of the present invention. In step 1501, one of two values (either a first or a second programmed value) is stored in a first memory cell. The first and second programmed values are different from each other. For example, in some embodiments the first programmed value is a high value and the second programmed value is a low value. In other embodiments, the first programmed value is a low value and the second programmed value is a high value. In step 1502, two values (a first and a second stored value) are stored in second and third memory cells.

**[0066]** In step 1503, when the first memory cell stores the first programmed value (e.g., a high value in the pictured embodiments), the first stored value is provided via a first signal path. In step 1504, when the first memory cell stores the second programmed value (e.g., a low value in the pictured embodiments), both the first and the second stored values are provided. The first stored value is provided via the first signal path, and the second stored value is provided via a second signal path.

**[0067]** Clearly, to avoid contention while providing both the first and the second stored values to the same node or output terminal, the first and second stored values should be the same. Also, the first and second signal paths are different for at least a portion of the two signal paths.

However, other portions of the two signal paths can traverse the same signal lines and logical elements.

**[0068]** In some embodiments, the steps of the method are performed using a PLD, e.g., an FPGA. In these embodiments, the first memory cell and/or the second and third memory cells can be configuration memory cells of the PLD. Thus, steps 1501 and 1502 can comprise configuring the PLD to store the programmed and/or stored values in the memory cells.

**[0069]** Fig. 16 illustrates the steps of a method of providing a logical value in a PLD, according to an embodiment of the present invention. The logical value is addressed by first and second signal lines (e.g., INx in the pictured embodiments). In step 1601, one of two different values (either a first or a second programmed value) is stored in a first memory cell. In some embodiments, the first programmed value is a high value and the second programmed value is a low value. In other embodiments, the first programmed value is a low value and the second programmed value is a high value. In step 1602, four values are stored in four additional memory cells.

**[0070]** In step 1603, a signal on the second signal line is utilized to select two signals comprising two of the four stored values. In step 1604, when the first memory cell stores the first programmed value (e.g., a high value in the pictured embodiments), exactly one of the two signals selected in step 1603 is selected. The selection is made based on a value stored on the first signal line. In step 1605, when the first memory cell stores the second programmed value (e.g., a low value in the pictured embodiments), both of the signals selected in step 1603 are selected irregardless of the value on the first signal line. To avoid contention on the node or output terminal to which the two signals are provided, both signals should have the same logical value when step 1605 applies.

**[0071]** In some embodiments, the first memory cell and/or the additional memory cells are configuration memory cells of

the PLD. Thus, steps 1601 and 1602 can comprise configuring the PLD to store the programmed and/or stored values in the memory cells. In some embodiments, the PLD is an FPGA.

**[0072]** In some embodiments, step 1602 is omitted. Instead of storing the four stored values in four additional memory cells, the four stored values are provided by four multiplexer circuits, with the values being stored prior to being provided to the multiplexer circuits (e.g., see Figs. 11-14). In some embodiments, some of the four stored values are provided by memory cells and at least one of the stored values is provided by a multiplexer circuit.

**[0073]** Fig. 17 illustrates the steps of a method of providing a logical value from a LUT, according to an embodiment of the present invention. The LUT includes a plurality of stored values and is controlled by a plurality of signal lines.

**[0074]** In step 1701, a PLD is configured with a PLD design. The PLD design can be, for example, an FPGA design. Configuring the PLD includes programming a configuration memory cell of the PLD with a first programmed value. The first programmed value indicates that one of the signal lines in the PLD design is used. In step 1702, another PLD memory cell is configured with a second programmed value. The second programmed value indicates that another one of the signal lines in the PLD design is unused. Clearly, step 1702 is only carried out when at least one of the signal lines in the PLD design is unused.

**[0075]** In step 1703, when all of the signal lines controlling the LUT are used in the PLD design, the LUT provides exactly one of the stored values. In step 1704, when at least one of the signal lines controlling the LUT is unused in the PLD design, the speed of the LUT is increased by providing more than one of the stored values. All of the more than one stored values have identical logical values.

**[0076]** In the exemplary LUT circuits and methods illustrated herein, the number of stored values is either

four or sixteen, the number of signal lines is either two or four, and the number of stored values provided in step 1704 is two, four, or eight. For example, when there are sixteen stored values and four signal lines, one unused signal line results in two stored values being provided, two unused signal lines results in four stored values being provided, and three unused signal lines results in eight stored values being provided. However, the illustrated circuits are purely exemplary. LUTs having other numbers of stored values, signal lines, and provided values are easily extrapolated from the examples provided herein. For example, one LUT circuit according to the invention includes sixty-four stored values addressed by six signal lines, and can simultaneously provide (depending on the number of unused signal lines) either one, two, four, eight, sixteen, or thirty-two stored values.

**[0077]** Those having skill in the relevant arts of the invention will now perceive various modifications and additions that can be made as a result of the disclosure herein. For example, the above text describes the circuits of the invention in the context of programmable logic devices (PLDs) such as field programmable logic devices (FPGAs). However, the circuits of the invention can also be implemented in other programmable devices, circuits, and systems.

**[0078]** Further, pullups, transistors, P-channel transistors, N-channel transistors, multiplexers, multiplexer circuits, passgates, CMOS passgates, memory cells, configuration memory cells, inverting logic gates, inverters, logical NAND gates, NAND gates, buffers, output buffers, and other components other than those described herein can be used to implement the invention. Moreover, some components are shown directly connected to one another while others are shown connected via intermediate components. In each instance the method of interconnection establishes some desired electrical communication between two or more circuit

nodes. Such communication can often be accomplished using a number of circuit configurations, as will be understood by those of skill in the art.

**[0079]** Accordingly, all such modifications and additions are deemed to be within the scope of the invention, which is to be limited only by the appended claims and their equivalents.